สุทธิศักดิ์ พงศ์ธนาพาณิช

Advanced BASIC Programming Technique ตอน การเขียนโปรแกรมสำหรับวินโดวส์ 95 ด้วย VB/Win 4.0-32 บิต

แนะนำเทคนิคและให้ข้อคิดเห็น ในการเขียนโปรแกรมสำหรับวินโดวส์ 95 ด้วย VB/Win 4.0-32 บิต

จากตอนที่แล้ว ผมได้แจ้งให้ผู้อ่านได้ทราบแล้วว่าตั้งแต่บทความตอนนี้เป็นต้นไป ผู้เขียนก็จะเขียนบทความเกี่ยวกับ VB/Win เฉพาะสำหรับวินโดวส์ 95 หรือ NT เท่านั้น ซึ่งจะเน้นหนักเฉพาะเทคนิคการโปรแกรมแอพพลิเคชันแบบ 32 บิต และเพื่อให้ผู้อ่านบทความนี้เป็นประจำมีพื้นฐานความเข้าใจเกี่ยวกับการโปรแกรมแอพพลิเคชันแบบ 32 บิต ด้วย VB/Win ผู้เขียนก็จะขอเริ่มต้นด้วยการแนะนำหลักการเบื้องต้นและมาตรฐานที่ถูกกำหนดโดยไมโครซอฟท์ในการเขียนแอพพลิเคชันสำหรับวินโดวส์ 95 โดยเนื้อหาที่ผู้เขียนอธิบายในบทความตอนนี้ ก็รวบรวมจากจดหมายของผู้อ่านบทความต่าง ๆ ของผู้เขียนในวารสารไมโครฯ ประมาณสัก 15 ฉบับ ซึ่งผู้อ่านกลุ่มนี้ได้ใช้ VB/Win 3.0 ในการเขียนโปรแกรมและต้องการที่จะหันมาใช้ VB/Win 4.0-32 บิต ในการสร้างแอพพลิเคชันสำหรับวินโดวส์ 95 ดังนั้นเพื่อให้แอพพลิเคชันที่ถูกสร้างเป็นไปตามข้อแนะนำ (หรือจริง ๆ อาจเรียกได้ว่าเป็นมาตรฐาน) ของไมโครซอฟท์ ซึ่งเป็นที่ยอมรับกันในหมู่โปรแกรมเมอร์ในต่างประเทศ และกลุ่มที่พัฒนาแอพพลิเคชันโดยใช้ซอฟต์แวร์ต่าง ๆ ของไมโครซอฟท์ (Solution Developer) ผมจึงได้รวบรวมแนวคิดและข้อแนะนำจากแหล่งต่าง ๆ ลงในบทความตอนนี้ และผู้เขียนหวังว่าผู้อ่านจะสามารถนำความรู้ที่ได้ไปใช้สร้างแอพพลิเคชันสำหรับวินโดวส์ 95 ได้อย่างสมบูรณ์และได้มาตรฐานเดียวกับแอพพลิเคชันระดับแนวหน้าที่เขียนขึ้นเพื่อการค้าโดยตรงนะครับ

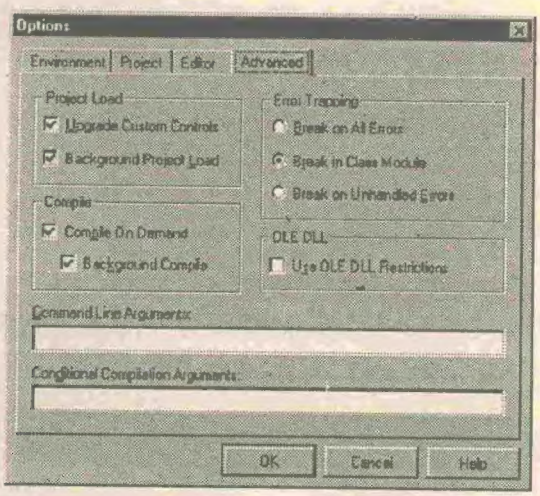
เนื้อหาที่ปรากฏในเอกสารฉบับนี้ อาจจะไม่สามารถเขียนให้ครอบคลุมคุณสมบัติใหม่ ๆ ของ VB/Win 4.0-32 บิต อย่างละเอียดได้ทั้งหมด ทั้งนี้เนื่องจากวัตถุประสงค์ของเอกสารนี้ก็เพื่อเป็นการแนะนำเทคนิคและให้ข้อคิดเห็นในการออกแบบการอินเทอร์เฟส คุณสมบัติที่สนับสนุน และเทคนิคการสร้างแอพพลิเคชันเท่านั้น ซึ่งถ้าหากผู้อ่านท่านใดมีข้อคิดเห็นที่คิดว่าจะเป็นประโยชน์กับผู้อ่านท่านอื่น ๆ ก็ขอให้เขียนจดหมายมาคุยกันได้ครับ เพื่อที่ผมจะได้นำเนื้อหามาเพิ่มเติมในบทความตอนต่อ ๆ ไป

เบื้องต้นเกี่ยวกับแอพพลิเคชันสำหรับวินโดวส์ 95

ในช่วงปลายปีพ.ศ. 2538 หลังจากที่ไมโครซอฟท์ได้เปิดตัววินโดวส์ 95 ซึ่งเป็นระบบปฏิบัติการระดับ 32 บิต (แต่นักสังเกตการณ์ส่วนใหญ่จะบอกว่ายังไม่ใช่ระบบปฏิบัติการระดับ 32 บิต ที่สมบูรณ์แบบมากนัก) VB/Win ก็ได้รับการพัฒนาขึ้นเป็นเวอร์ชัน 4.0 ซึ่งมีทั้งสำหรับการพัฒนาโค้ดแบบ 16 บิตและ 32 บิต และเป็นตัวแปลภาษาที่ได้รับการคาดหวังว่าจะสามารถใช้สร้างแอพพลิเคชันสำหรับวินโดวส์ 95 ได้อย่างรวดเร็ว และเนื่องจากวินโดวส์ 95 เป็นระบบปฏิบัติการที่ไมโครซอฟท์ใช้ในการเปิดทางไปสู่การทำงานร่วมกับ OLE อย่างค่อนข้างชัดเจน ซึ่งผู้อ่านสามารถสังเกตได้จากไฟล์คอนโทรล .OCX หรือความสามารถในการสร้าง OLE Server หรือการสร้างไฟล์ OLE DLL เป็นต้น ดังนั้นแอพพลิเคชันที่สามารถใช้โลโกสำหรับวินโดวส์ 95 ได้นั้น จะต้องมีการออกแบบส่วนต่าง ๆ เช่น ลักษณะการใช้สี ส่วนอินเทอร์เฟสกับผู้ใช้ หรือ OLE เป็นต้น ให้มีความเป็นมาตรฐานเดียวกัน เพื่อให้ได้รับการยอมรับจากไมโครซอฟท์ โดยที่ข้อแนะนำพื้นฐานดังกล่าวข้างต้นมีรายละเอียดดังต่อไปนี้

แอพพลิเคชันต้องเป็น 32 บิต

เพื่อให้แอพพลิเคชันสามารถใช้ความสามารถจากโค้ดส่วน 32 บิต ของวินโดวส์ 95 ได้อย่างสมบูรณ์ ดังนั้นผู้อ่านต้องเปลี่ยนไฟล์คอนโทรลจาก VBX มาเป็น .OCX (OLE Control Extension) และที่สำคัญถ้าหากโค้ดส่วนต่าง ๆ ภายในแอพพลิเคชันมีการเรียกใช้ฟังก์ชันวินโดวส์ API ผู้อ่านก็ต้องแทนที่กลุ่มฟังก์ชันดังกล่าวด้วยฟังก์ชัน Win32 โดยเมื่อผู้อ่านทำการคอมพายล์โค้ดภายใต้วินโดวส์ 95 หรือวินโดวส์ NT 3.51 ด้วย VB/Win 4.0-32 บิต ก็จะได้แอพพลิเคชันระดับ 32 บิตตามที่ต้องการทันที ซึ่งผู้เขียนขอสรุปเป็นขั้นตอนต่าง ๆ ได้ดังต่อไปนี้



รูปที่ 1 แสดงไดอะล็อกบ็อกซ์ของคำสั่ง Options ของเมนู Tools

- ใช้คอนโทรล OLE (.OCX) ที่เป็นคอนโทรลระดับ 32 บิต แทนคอนโทรล .VBX ซึ่งเป็นคอนโทรลระดับ 16 บิต โดยที่ผู้อ่านสามารถกำหนดให้ VB/Win จัดการให้โดยอัตโนมัติได้โดยการเลือกรายการ Upgrade Custom Controls ในกลุ่ม Project Load ในหัวข้อ Advanced ของไดอะล็อกบ็อกซ์ Options (จากคำสั่ง Options ของเมนู Tools) ดังในรูปที่ 1 โดยเมื่อผู้อ่านโหลดไฟล์โปรเจ็กต์จากเวอร์ชัน 3.0 VB/Win ก็จะทำการค้นหาไฟล์ .OCX ที่เหมือนกับไฟล์ .VBX และจะแสดงผลไดอะล็อกบ็อกซ์เพื่อให้ผู้ใช้เลือกที่จะใช้ไฟล์คอนโทรลเดิม (.VBX) หรือไฟล์คอนโทรลใหม่ (.OCX) โดยถ้าหากผู้ใช้ตอบตกลง VB/Win ก็จะแทนที่ไฟล์คอนโทรล .VBX ด้วยไฟล์คอนโทรล .OCX ที่เหมือนกันทันทีโดยอัตโนมัติ (การจับคู่ไฟล์ .VBX และไฟล์ .OCX ที่เหมือนกันจะถูกกระทำโดยอัตโนมัติในขณะที่ผู้อ่านติดตั้ง VB/Win 4.0)

ส่วนวิธีการเพิ่มไฟล์ OLE control ให้กับโปรเจ็กต์นั้น ก็แตกต่างไปจาก VB/Win 3.0 โดยผู้อ่านจะเห็นว่าสำหรับคำสั่ง Add File ของ VB/Win 3.0 นั้นเราสามารถใช้เพิ่มไฟล์ .VBX ให้กับโปรเจ็กต์ได้ทันที แต่สำหรับ VB/Win 4.0-32 บิต ไม่สนับสนุนการเพิ่มไฟล์ .OCX ให้กับโปรเจ็กต์ด้วยคำสั่งเดียวกันนี้ แต่ผู้อ่านสามารถกระทำได้โดยการใช้คำสั่ง Custom Controls ของเมนู Tools โดยเมื่อผู้อ่านเลือกคำสั่งนี้ก็จะปรากฏไดอะล็อกบ็อกซ์ Custom Controls ซึ่งผู้อ่านสามารถเลือกคอนโทรลที่ต้องการเพิ่มให้กับโปรเจ็กต์ได้ทันที โดยตัวอย่างในรูปที่ 2 เป็นการกำหนดคอนโทรล Apex Data Bound Grid และคอนโทรล Microsoft Common Dialog Control ให้กับโปรเจ็กต์ ที่สำคัญผู้อ่านสามารถเลือกที่จะแทรก Insertable Object ให้กับแอพพลิเคชันได้อีกด้วย (Insertable Object จะปรากฏในไดอะล็อกบ็อกซ์ก็ต่อเมื่อรายการ Insertable Objects ถูกเลือกเท่านั้น) สำหรับ Insertable Object นั้นก็หมายถึง ออบเจ็กต์ใด ๆ ของแอพพลิเคชันที่สามารถถูกเรียกใช้ได้โดยแอพพลิเคชันอื่น ๆ ซึ่งออบเจ็กต์นี้ก็จะเป็นออบเจ็กต์ประเภทเดียวกับ custom control เช่น Word Basic ของไมโครซอฟต์เวิร์ด หรือ Sheet ของไมโครซอฟต์เอ็กเซล เป็นต้น นอกจากนี้แอพพลิเคชันของบริษัทอื่น ๆ ที่สนับสนุน OLE Automation ก็สามารถเป็น Insertable Object ให้กับ VB/Win ได้เหมือนกัน

หมายเหตุ สำหรับไฟล์ .OCX ที่มากับ VB/Win 4.0 นั้นจะมีทั้งชนิดที่เป็นคอนโทรลระดับ 16 บิต และ 32 บิต ซึ่งผู้อ่านสามารถสังเกตได้จากชื่อของไฟล์ โดยถ้าหากชื่อไฟล์ลงท้ายด้วยตัวเลข 16 เช่น GRID16.OCX ก็จะเป็นไฟล์คอนโทรลระดับ 16 บิต แต่ถ้าหากชื่อไฟล์ลงท้ายด้วยตัวเลข 32 เช่น GRID32.OCX ก็จะเป็นไฟล์คอนโทรลระดับ 32 บิต เป็นต้น แต่ตัวอย่างการตั้งชื่อไฟล์คอนโทรลข้างต้นก็อาจ

จะไม่แน่เสมอไป เพราะบางคอนโทรลก็ไม่มีการกำหนดตัวเลขระบุระดับเช่นกัน เช่น SYSINFO.OCX ซึ่งเป็นไฟล์คอนโทรลระดับ 32 บิต เป็นต้น

- โหลดไฟล์โปรเจ็กต์ (.MAK) ที่ถูกจัดเก็บในรูปแบบของจากเวอร์ชัน 2.0 หรือ 3.0 ซึ่งเมื่อ VB/Win 4.0-32 บิต โหลดไฟล์โปรเจ็กต์ดังกล่าว ก็จะปรากฏไดอะล็อกบ็อกซ์เพื่อถามผู้ใช้ และแนะนำให้จัดเก็บไฟล์ทั้งหมดรวมทั้งไฟล์โปรเจ็กต์ในรูปแบบของเวอร์ชัน 4.0 (ไฟล์โปรเจ็กต์ของVB/Win 4.0-32 บิต ผู้อ่านสามารถจัดเก็บได้โดยใช้นามสกุล .VBP หรือ .MAK ก็ได้ แต่ไฟล์นามสกุล .VBP จะถือว่าเป็นนามสกุล default ของ VB/Win 4.0) ดังในรูปที่ 3 เป็นไดอะล็อกบ็อกซ์ที่จะปรากฏเมื่อผู้อ่านโหลดไฟล์โปรเจ็กต์ที่ถูกจัดเก็บในรูปแบบเวอร์ชันก่อนหน้านี้ (1.0, 2.0 และ 3.0) โดยถ้าหากผู้อ่านเลือกปุ่มคำสั่ง OK VB/Win4.0-32 บิตก็จะจัดเก็บในรูปแบบเวอร์ชัน 4.0 เฉพาะฟอร์มที่ปรากฏชื่อในไดอะล็อกบ็อกซ์เท่านั้น แต่ถ้าหากผู้อ่านเลือกปุ่มคำสั่ง Ok For All VB/Win 4.0-32 บิต ก็จะจัดเก็บในรูปแบบเวอร์ชัน 4.0 ทุก ๆ ฟอร์มของโปรเจ็กต์นั้น ๆ สำหรับไฟล์ฟอร์มที่จัดเก็บในรูปแบบของ VB/Win 4.0 โค้ดบรรทัดแรกจะเป็นการระบุเวอร์ชันดังนี้ VERSION 4.00

- ใช้ฟังก์ชันวินโดว์ API ระดับ 32 บิต สำหรับแอพพลิเคชันที่มีการเรียกใช้ฟังก์ชันเหล่านี้ โดยถ้าหากผู้อ่านมีการเรียกใช้ฟังก์ชันวินโดว์ API ระดับ 16 บิต ใน VB/Win 4.0-32 บิต ก็จะเกิดข้อผิดพลาดทันที แต่ถ้าหากผู้อ่านต้องการคงประโยคการประกาศหรือประโยคที่เรียกใช้ฟังก์ชันวินโดว์ API ระดับ 16 บิตไว้ในซอร์สโค้ด เพื่อให้โค้ดคอมแพตทิเบิลทั้ง 16 บิต และ 32 บิต ก็สามารถหลีกเลี่ยงข้อผิดพลาดในขณะคอมพายล์ได้ โดยการใช้ประโยคการคอมพายล์อย่างมีเงื่อนไข

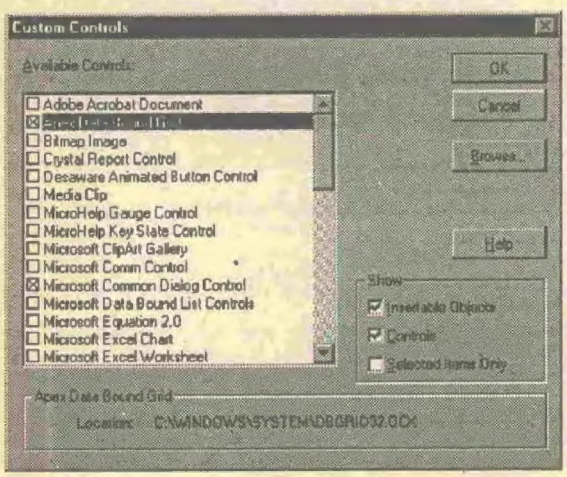
```
#If Win32 Then
Declare Function SendMessage Lib "User32" Alias "SendMessageA" ( _
ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, _
lParam As Long) As Long
#Else
Declare Function SendMessage Lib "User" (ByVal hWnd As Integer, _
ByVal wMsg As Integer, ByVal wParam As Integer, lParam As Any) _
As Long
#End If
```

(#If...Then...#Else) พร้อมทั้งการตรวจสอบค่าของค่าคงที่ของระบบ Win32 ดังเช่น

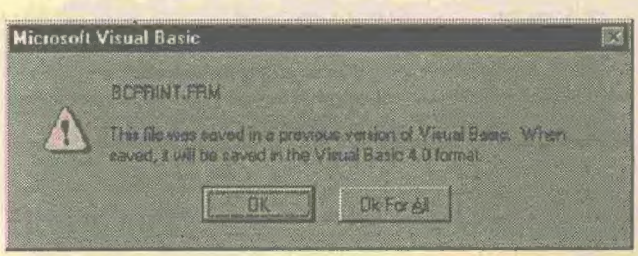
สำหรับโค้ดข้างต้น เมื่อผู้อ่านทำการคอมพายล์โปรแกรม VB/Win จะคอมพายล์โค้ดในส่วน #If Win32 Then... ถ้าหากคอมพายล์ด้วย VB/Win 4.0-32 บิต และในทางกลับกันก็จะคอมพายล์โค้ดในส่วน #Else...End If ถ้าหากคอมพายล์ด้วย VB/Win 4.0-16 บิต เป็นต้น สำหรับคำสงวน Alias ที่ปรากฏในประโยคการประกาศข้างต้นนั้น ถูกใช้ในการประกาศให้ฟังก์ชันวินโดว์ API ที่ชื่อ SendMessageA จะถูกแทนที่ด้วยชื่อ SendMessage สำหรับ VB/Win ทั้งนี้ก็เพื่อให้มีชื่อเหมือนกับเวอร์ชัน 16 บิตนั่นเอง

ซึ่งผู้อ่านจะเห็นว่าสำหรับวินโดวส์ 3.1x ฟังก์ชัน SendMessage จะอยู่ในโมดูล User แต่สำหรับวินโดวส์ 95 ฟังก์ชัน SendMessageA จะถูกจัดเก็บไว้ในโมดูล User32 ซึ่งเป็นโมดูลที่ให้บริการด้านเดียวกันเพียงแต่เป็นโค้ดระดับ 16 บิต และ 32 บิตตามลำดับเท่านั้น แต่ก็มีบางฟังก์ชันที่มีการย้ายจากโมดูลหนึ่งไปยังอีกโมดูลหนึ่งไปเลยก็มี และเพื่อความมั่นใจผู้เขียนก็ขอแนะนำให้ผู้อ่านคัดลอกประโยคการประกาศฟังก์ชันเหล่านี้โดยใช้โปรแกรม Text API Viewer ที่มากับ VB/Win 4.0 จะสะดวกกว่ามาก ซึ่งโปรแกรมนี้สามารถอ่านฐานข้อมูลได้ทั้งชนิดที่เป็นไฟล์แอสกีและไฟล์ฐานข้อมูลของ Microsoft Access (.MDB) แต่โปรแกรมจะสามารถจัดการกับไฟล์ที่จัดเก็บในรูปแบบ .MDB ได้รวดเร็วกว่า

สำหรับการประกาศฟังก์ชัน ค่าคงที่ หรือตัวแปรโครงสร้างสำหรับฟังก์ชัน API ทั้งระดับ 16 บิต และ 32 บิต สำหรับ VB/Win 4.0 นั้นผู้อ่านสามารถคัดลอกมายังเอนไวรอนเมนท์ได้โดยการใช้โปรแกรมที่ติดมากับ VB/Win 4.0 ที่ชื่อ Text API Viewer ดังเช่นตัวอย่างในรูปที่ 4 ผู้เขียนได้ใช้โปรแกรมดังกล่าวในการเปิดไฟล์แอสกีที่ชื่อ



รูปที่ 2 แสดงไดอะล็อกบ็อกซ์ Custom Controls



รูปที่ 3 แสดงไดอะล็อกบ็อกซ์เมื่อโหลดโปรเจ็กต์เวอร์ชัน 2.0 หรือ 3.0

win32api.txt ซึ่งเป็นไฟล์ที่จัดเก็บฟังก์ชัน ค่าคงที่ หรือตัวแปรโครงสร้างสำหรับฟังก์ชันวินโดวส์ระดับ 32 บิต (วินโดวส์ 95 และ NT 3.5x) เป็นต้น

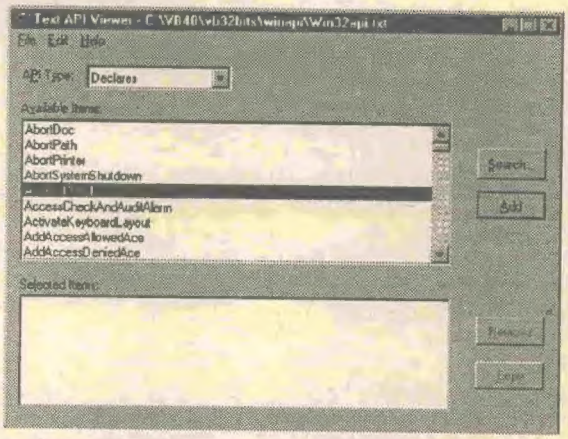
หมายเหตุ ฟังก์ชัน API ระดับ 32 บิต ของวินโดวส์ 95 และวินโดวส์ NT 3.51 นั้นในปัจจุบันยังคงมีความแตกต่างกันอยู่ (อย่างน้อยก็หน้าที่ของฟังก์ชัน) แต่ในเวอร์ชันถัดไปของวินโดวส์ (อาจจะเป็นวินโดวส์ 97 และ NT 4.0) ความแตกต่างดังกล่าวก็จะถูกขจัดไปทั้งหมด ซึ่งเมื่อถึงจุดนั้นผู้เขียนก็ยังมองไม่ค่อยออกว่าวินโดวส์ 95 และวินโดวส์ NT สำหรับเวอร์คสเตชันนั้นจะมีความแตกต่างกันอย่างไร

สำหรับรายละเอียดเกี่ยวกับวิธีการใช้ การประกาศหรือความหมายของฟังก์ชันวินโดว์ API (Win32) ผู้อ่านสามารถศึกษาได้จากชุดหนังสือ Windows 95 SDK ซึ่งจัดพิมพ์โดยบริษัทไมโครซอฟท์ หนังสือชุดนี้มีความสำคัญอย่างมาก ดังนั้นโปรแกรมเมอร์ทุกท่านควรจะใช้เป็นหนังสืออ้างอิงสำหรับการเขียนโปรแกรม เพราะฟังก์ชัน API ทั้งหมดสำหรับวินโดวส์ 95 ได้ถูกรวบรวมเอาไว้ในหนังสือชุดนี้อย่างสมบูรณ์

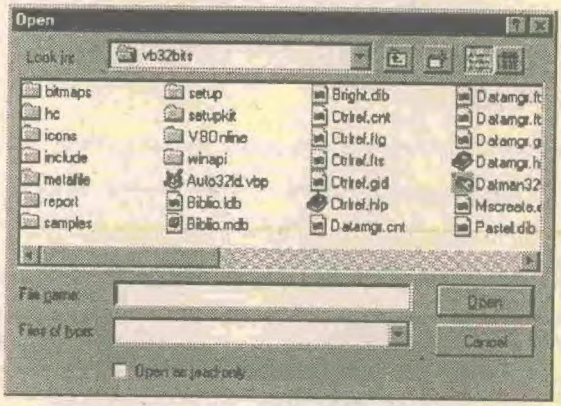
- คอมพายล์โปรเจ็กต์เสียใหม่ ซึ่งถ้าหากผู้อ่านทำการคอมพายล์ภายใต้วินโดวส์ 95 ด้วย VB/Win 4.0-32 บิต VB/Win ก็จะทำการสร้างไฟล์ .EXE ระดับ 32 บิต โดยอัตโนมัติทันที ซึ่งแอพพลิเคชันระดับ 32 บิตนี้สามารถทำงานได้ทั้งภายใต้วินโดวส์ 95 และวินโดวส์ NT 3.51 แต่โค้ด 32 บิตที่สร้างโดย VB/Win 4.0-32 บิต นี้ยังคงไม่สนับสนุนการทำงานแบบมัลติเทรด (multithread) ดังนั้นไฟล์ .EXE ที่คอมพายล์ด้วย VB/Win 4.0-32 บิต จึงไม่สามารถใช้ความสามารถด้านการทำงานแบบมัลติเทรดของวินโดวส์ NT ได้เลย ซึ่งผู้เขียนคาดว่าในเวอร์ชันถัดไปของ VB/Win (เวอร์ชัน 5.0) จะสามารถกำจัดของจำกัดนี้ออกไปได้อย่างสมบูรณ์ เพื่อเป็นการเพิ่มสมรรถนะของแอพพลิเคชันได้มากทีเดียว

สนับสนุนการตั้งชื่อไฟล์มากถึง 255 ตัวอักษร

แอพพลิเคชันต้องสนับสนุนการตั้งชื่อไฟล์ตามความสามารถใหม่ของวินโดวส์ 95 หรือวินโดวส์ NT 3.51 ซึ่งไม่เหมือนกับชื่อไฟล์ในระบบ 8.3 ตัวอักษรของดอส นั่นก็คือวินโดวส์ 95 สนับสนุนการตั้งชื่อไฟล์และไดเรคทอรี (โฟลเดอร์) มากถึง 255 ตัวอักษร และยังอนุญาตให้มีการใช้ช่องว่าง (ascii 32) ในการตั้งชื่อได้อีกด้วย ดังนั้นถ้าหากแอพพลิเคชันของผู้อ่านไม่ต้องการไดอะล็อกบ็อกซ์สำหรับการเปิดหรือจัดเก็บไฟล์ที่เฉพาะเจาะจง ผู้เขียนก็ขอแนะนำให้ผู้อ่านใช้ไดอะล็อกบ็อกซ์มาตรฐานของวินโดวส์ 95 โดยการใช้คอนโทรล Common Dialog (COMDLG32.OCX) ซึ่งเป็นคอนโทรลที่สนับสนุนไดอะล็อกบ็อกซ์มากมาย ดังนี้ Open, Save As, Font, Color และ Print เป็นต้น โดยในการเขียนโค้ดผู้อ่านสามารถกำหนดให้ไดอะล็อกบ็อกซ์ใด ๆ ปรากฏได้โดยการกำหนดค่าให้กับวิธี Show... เช่น ประโยคข้างล่างจะแสดงผลไดอะล็อกบ็อกซ์ Open และกำหนดค่าต่าง ๆ ตามค่าปกติของคอนโทรล Common Dialog ดังในรูปที่ 5



รูปที่ 4 แสดงตัวอย่างการใช้ Text API Viewer



รูปที่ 5 แสดงไดอะล็อกบ็อกซ์ Open ของคอนโทรล Common Dialog

```
CMDialog1.ShowOpen
```

จากรูปที่ 5 ผู้อ่านจะเห็นว่าเราไม่สามารถที่จะสร้างไดอะล็อกบ็อกซ์ Open ได้เลยด้วยโค้ดตามที่เคยใช้ใน VB/Win เวอร์ชัน 3.0 ทั้งนี้เนื่องจากวินโดวส์ 95 จะจัดการกับการแสดงผลไอคอนตามชนิดของไฟล์ด้วยตัวเองโดยอัตโนมัติ ซึ่งชนิดของไฟล์นี้ผู้อ่านสามารถกำหนดให้กับวินโดวส์ 95 ได้ด้วยตัวเอง และที่สำคัญอีกประการเพราะว่าวินโดวส์ 95 สนับสนุนชื่อไฟล์มากถึง 255 ตัวอักษร ดังนั้นจึงทำให้ยากที่จะตรวจสอบว่าชื่อของไฟล์นั้นถูกต้องตามกฎหรือไม่ ตามที่เคยใช้กับการตั้งชื่อไฟล์ 8.3 ของดอส ดังนั้นวิธีที่ง่ายที่สุดในการตรวจสอบความถูกต้องของชื่อไฟล์ ผู้เขียนก็ขอแนะนำให้ผู้อ่านใช้วิธีการตรวจจับข้อผิดพลาด (error handling routine) เช่นในโค้ดตัวอย่างต่อไปนี้ ถ้าหากเกิดข้อผิดพลาดขึ้นมา VB/Win ก็จะกระโดดไปยังโค้ดในส่วนของ ErrTrap: เป็นต้นไปและในที่นี้ผู้เขียนจะตรวจสอบหมายเลขข้อผิดพลาดโดยถ้าหากมีค่าเท่ากับ 76 จะแสดงว่าการตั้งชื่อไฟล์ไม่ถูกต้องหรือไดเรคทอรีที่กำหนดไม่มีอยู่จริงนั่นเอง

```
Private Sub SaveMasterFile(filename$)

    ' Turn on error handling.
    On Error GoTo Errtrap
    ' Test if valid filename.
    Open filename$ For Binary As #1
    ' Operation code begine from here.
    .
    .
    .

ErrReturn:
    Close #1
    Exit Sub

Errtrap:
    If Err.Number = 76 Then
      MsgBox "Path not found or file name is_
invalid."
    End If
    Resume ErrReturn
End Sub
```

การเลือกใช้สี

ในการเลือกสีสำหรับการแสดงผลฟอร์มหรือคอนโทรล ให้ผู้อ่านเลือกใช้สีพื้นฐานของระบบ (ลำดับที่ 0 ถึง 15) เช่น สมมติว่าผู้อ่านทำการออกแบบไอคอน ในการให้สีส่วนต่าง ๆ ของไอคอน ก็ควรจะเลือกใช้สีใด ๆ ใน 16 สีพื้นฐานนี้ เป็นต้น ทั้งนี้ก็เพื่อให้แอพพลิเคชันที่ได้มีความคล้ายคลึงกันแอพพลิเคชันอื่น ๆ ที่ทำงานภายใต้วินโดวส์ 95 ซึ่งจะส่งผลให้ผู้ใช้มีความคุ้นเคยและมีความเข้าใจต่อความหมายของสีต่าง ๆ ง่ายขึ้น เช่น ผู้อ่านสามารถดูตัวอย่างได้จากเมนู ซึ่งโดยปกติคำสั่งต่าง ๆ ของเมนูจะแสดงผลด้วยตัวอักษรสีดำ และถ้าหากคำสั่งใดไม่สามารถใช้ได้ในขณะนั้น สีของตัวอักษรที่ใช้แสดงผลก็จะกลายเป็นสีขาวสลับเทาหรือที่เรียกว่า dimmed เป็นต้น และสำหรับสีมาตรฐานของระบบที่ถูกใช้ในการแสดงผลส่วนต่าง ๆ โดยวินโดวส์ 95 นั้น VB/Win ได้ประกาศค่าคงที่เอาไว้ดังตารางที่ 1

และนอกจากนี้โปรแกรมเมอร์มืออาชีพยังแนะนำให้มีไอคอนทั้งขนาดเล็กและขนาดใหญ่ สำหรับทูลบาร์ทุก ๆ แอพพลิเคชัน สำหรับผู้อ่านที่เคยใช้ VB/Win 3.0 จะเห็นว่าไอคอนที่สามารถกำหนดให้กับแอพพลิเคชันนั้น จะมีเฉพาะไอคอนขนาดใหญ่ (32 x 32 จุด) เท่านั้น แต่สำหรับแอพพลิเคชันสำหรับวินโดวส์ 95 นั้น ผู้อ่านสามารถกำหนดไอคอนขนาดเล็ก (16 x 16 จุด) ให้กับแอพพลิเคชันได้อีกด้วย ซึ่งในกรณีนี้แอพพลิเคชันจะทำการเลือกไอคอนสำหรับแสดงผลในเงื่อนไขต่าง ๆ โดยอัตโนมัติ ในการสร้างไอคอนทั้งสองขนาด ผู้อ่านสามารถกระทำได้โดยการใช้โปรแกรม Image Editor ซึ่งมากับ VB/Win 4.0 (โปรแกรม Image Editor มีเฉพาะชุด VB/Win ที่มากับ CD-ROM เท่านั้น)

ตารางที่ 1

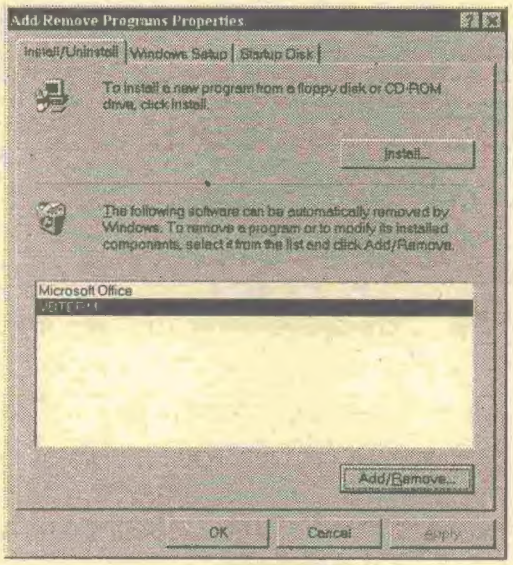
สัญลักษณ์	ค่าคงที่	รายละเอียด
vbScrollBars	0x80000000	สีของ Scroll bar
vbDesktop	0x80000001	สีของ Desktop
vbActiveTitleBar	0x80000002	สีของ title bar สำหรับวินโดวส์ที่ active
vbInactiveTitleBar	0x80000003	สีของ title bar สำหรับวินโดวส์ที่ inactive
vbMenuBar	0x80000004	สีพื้นของเมนู
vbWindowBackground	0x80000005	สีพื้นของวินโดวส์
vbWindowFrame	0x80000006	สีเฟรมของวินโดวส์
vbMenuText	0x80000007	สีตัวอักษรของเมนู
vbWindowText	0x80000008	สีตัวอักษรของวินโดวส์
vbTitleBarText	0x80000009	สีตัวอักษรของ caption
vbActiveBorder	0x8000000A	สีของ Border ของวินโดวส์ที่ active
vbInactiveBorder	0x8000000B	สีของ Border ของวินโดวส์ที่ inactive
vbApplicationWorkspace	0x8000000C	สีพื้นของแอพพลิเคชัน MDI
vbHighlight	0x8000000D	สีพื้นของรายการที่ถูกเลือกของคอนโทรล
vbHighlightText	0x8000000E	สีตัวอักษรของรายการที่ถูกเลือกของคอนโทรล
vbButtonFace	0x8000000F	สีระบายพื้นของ command button
vbButtonShadow	0x80000010	สีระบายขอบของ command button
vbGrayText	0x80000011	สีแสดงตัวอักษร dimmed
vbButtonText	0x80000012	สีตัวอักษรของ push button
vbInactiveCaptionText	0x80000013	สีตัวอักษรของ caption ที่อยู่ในสถานะ inactive
vb3DHighlight	0x80000014	สีสว่างสำหรับการแสดงผลแบบ 3 มิติ
vb3DDKShadow	0x80000015	สีเงาสำหรับการแสดงผลแบบ 3 มิติ
vb3DLight	0x80000016	สีสว่างที่สุดสำหรับการแสดงผลแบบ 3 มิติ
vbInfoText	0x80000017	สีตัวอักษรของ ToolTips
vbInfoBackground	0x80000018	สีพื้นของ ToolTip

สนับสนุนการติดตั้งและยกเลิกโปรแกรม

ตามแบบฉบับของแอพพลิเคชันขั้นนำทั่วไป ไมโครซอฟท์แนะนำให้โปรแกรม Setup ต้องสนับสนุนการยกเลิกโปรแกรม (uninstall) ด้วย เพื่อให้ผู้ใช้สามารถลบทิ้งไฟล์ต่าง ๆ ของแอพพลิเคชันได้อย่างสมบูรณ์ และที่สำคัญเนื่องจากในการติดตั้งวินโดวส์ 95 จะทำการลงทะเบียนแอพพลิเคชันเอาไว้ในไฟล์ system registry เพื่อให้แอพพลิเคชันสามารถเรียกใช้หรือให้บริการความสามารถด้าน OLE ได้อย่างถูกต้อง ดังนั้น ถ้าหากผู้อ่านทำการลบทิ้งแอพพลิเคชันดังกล่าว ก็จะต้องทำการลบทิ้งทะเบียนที่ลงไว้ในไฟล์ system registry ด้วยเช่นกัน สำหรับ VB/Win 4.0-32 บิต นั้นผู้อ่านสามารถกระทำได้อย่างง่ายดายโดยการใช้ Setup Wizard-32 บิต ซึ่งผู้อ่านจะเห็นว่าการเขียนโปรแกรมติดตั้งที่ทำการติดตั้งแอพพลิเคชันเวอร์ชันใหม่และทำการลบทิ้งแอพพลิเคชันเก่าทิ้งไปนั้น สำหรับวินโดวส์ 3.1x นั้นทำได้ค่อนข้างยากทีเดียว โดยเฉพาะกับแอพพลิเคชันที่เขียนด้วย VB/Win 3.0 เพราะ VB/Win 3.0 ไม่สนับสนุนการกำหนดเวอร์ชันของไฟล์แอพพลิเคชัน (version stamp support) จึงทำให้ไม่สามารถทำการตรวจสอบความทันสมัยของไฟล์ และที่สำคัญอีกประการ โปรแกรม Setup ที่สร้างด้วย Setup Wizard-32 บิตที่มากับ VB/Win 4.0-32 บิตนั้นยังสามารถลบทิ้งไฟล์ทั้งหมดที่ถูกติดตั้งโดยอัตโนมัติทันที ถ้าหากผู้ใช้มีการยกเลิกการติดตั้งในระหว่างทำการติดตั้ง ซึ่งความสามารถเหล่านี้ไม่สามารถสร้างได้ง่าย ๆ ด้วย VB/Win 3.0

ตามที่กล่าวมาแล้วว่าในขั้นตอนการติดตั้งแอพพลิเคชันวินโดวส์ 95 จะทำการลงทะเบียนแอพพลิเคชันไว้ในไฟล์ system registry ดังนั้นในการยกเลิกโปรแกรมผู้อ่านก็สามารถกระทำได้โดยการใช้คำสั่ง Add/Remove Program Properties ของ CONTROL PANEL ดังตัวอย่างในรูปที่ 6 เป็นการใช้คำสั่งดังกล่าวของ CONTROL PANEL ลบทิ้งโปรแกรม VBTERM ซึ่งเป็นแอพพลิเคชันตัวอย่างที่มากับ VB/Win 4.0

หมายเหตุ ถ้าหากผู้อ่านท่านใดที่สร้างแอพพลิเคชันทั้งระดับ 16 บิต และ 32 บิต ผู้เขียนขอแนะนำให้ผู้อ่านสร้างโปรแกรม Setup แยกต่างหากออกจากกัน ทั้งนี้เนื่องจากในขบวนการติดตั้งวินโดวส์จะทำการประมวลผลไฟล์ .EXE หรือไฟล์ .DLL ที่แตกต่างกัน ขึ้นกับว่าเป็นวินโดวส์ระดับ 16 บิต หรือ 32 บิตนั่นเอง โดยปกติโปรแกรม Setup สำหรับวินโดวส์ 95 มักจะชื่อ SETUP.EXE ส่วนโปรแกรม Setup สำหรับวินโดวส์ 3.xx ไม่กจะชื่อ SETUP16.EXE เป็นต้น แต่ถ้าหากโปรแกรม Setup ที่ถูกเขียนขึ้นมาเพื่อทำการคัดลอกซอร์สโค้ดหรืออื่น ๆ ที่ไม่เกี่ยวข้องกับแอพพลิเคชัน ผู้อ่านก็สามารถที่จะสร้างโปรแกรม Setup เพียงไฟล์เดียวสำหรับคัดลอกไฟล์ทั้งสำหรับวินโดวส์ระดับ 16 บิตและ 32 บิต (ซึ่งในกรณีนี้โปรแกรม Setup ต้องเป็นโค้ดระดับ 16 เท่านั้น เพราะโค้ดระดับ 16 บิต จะสามารถทำงานได้ทั้งบนวินโดวส์ 3.1x, 95 และ NT 3.51 นั่นเอง)



รูปที่ 6 แสดงวิธีการลบทิ้งแอพพลิเคชันด้วย CONTROL PANEL

สำหรับการเขียนโปรแกรมประเภท Uninstaller นั้น ผมขอแนะนำให้ผู้อ่านระวังในการลบทิ้งไฟล์ที่มีนามสกุลดังต่อไปนี้ .DLL, .OCX, .VBX เป็นต้น ทั้งนี้เนื่องจากไฟล์เหล่านี้เป็นไฟล์ไลบรารีที่สามารถถูกเรียกใช้งานพร้อม ๆ กันได้และอาจจะถูกเรียกใช้โดยแอพพลิเคชันอื่น ๆ ด้วยก็ได้ ดังนั้นก่อนลบทิ้งไฟล์เหล่านี้ ก็ควรที่จะมีการแสดงผลไดอะล็อกบ็อกซ์ เพื่อเป็นการถามย้ำผู้ใช้ว่า จะยังคงมีความต้องการที่จะลบทิ้งไฟล์เหล่านี้อีกหรือไม่ ซึ่งโดยปกติไฟล์ .DLL, .OCX, .VBX มักจะถูกจัดเก็บในไดเรคทอรีย่อย \SYSTEM ของวินโดวส์

ในความเป็นจริงในการเขียนโปรแกรมติดตั้งไฟล์ .VBX และ .OCX ของ VB/Win นั้น ผู้อ่านสามารถเลือกที่จะติดตั้งไฟล์เหล่านี้ในไดเรคทอรีย่อย \SYSTEM ของวินโดวส์ หรือในไดเรคทอรีของแอพพลิเคชันก็ได้ ซึ่งผู้อ่านจะสังเกตได้ว่า เมื่อมีการโหลดไฟล์โปรเจ็กต์ด้วยเอนไวรอนเมนท์ของ VB/Win ถ้าหากไฟล์คอนโทรลเหล่านี้อยู่ในไดเรคทอรีอื่น ๆ ที่ไม่ใช่ไดเรคทอรีย่อย \SYSTEM ของวินโดวส์ VB/Win ก็จะค้นหาไฟล์ดังกล่าวจากไดเรคทอรีที่กำหนดทันที

การสร้างไฟล์ .VBX และ .OCX นั้นไม่สามารถทำได้ด้วย VB/Win เพราะไฟล์คอนโทรลเหล่านี้เป็นไฟล์ไลบรารีชนิดไดนามิกลิงค์นั่นเอง ซึ่งมันจะถูกโหลดสู่หน่วยความจำทันทีที่แอพพลิเคชันที่เรียกใช้ไฟล์เหล่านี้ถูกเรียกสู่หน่วยความจำ ส่วนในการสร้างไฟล์เหล่านี้จะสามารถกระทำได้โดยใช้ Microsoft C/C++, Visual C++, Borland C หรือ MASM 6.11 ซึ่งผู้อ่านสามารถอ่านรายละเอียดเพิ่มเติมได้จากคู่มือของ VB/Win Professional

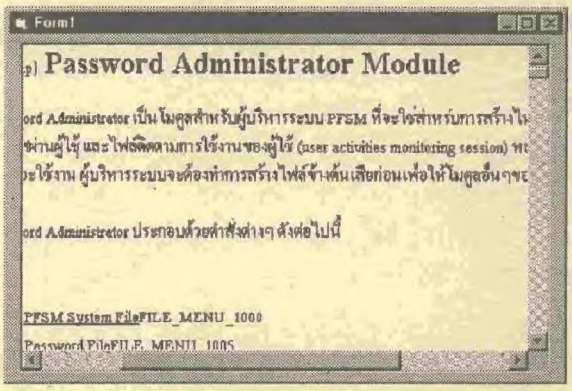
สนับสนุนการ drag & drop ระหว่างแอพพลิเคชัน

แอพพลิเคชันจะต้องสนับสนุนการ drag & drop ระหว่างแอพพลิเคชันโดยผ่านทาง OLE ซึ่งผู้อ่านสามารถกระทำได้โดยง่าย ๆ ด้วยการใช้คอนโทรล Rich Text เป็นต้น และใช้คอนโทรล OLE ในการเชื่อมโยงออบเจ็กต์ (คอนโทรล OLE จะทำหน้าที่เฉพาะ Client เท่านั้น ไม่เกี่ยวกับ OLE Server) สำหรับผู้อ่านที่ใช้คอนโทรล Text Box และ OLE ที่มากับ VB/Win 3.0 ก็สามารถแทนที่คอนโทรลเหล่านี้ด้วยคอนโทรล Rich Text และ OLE ที่มากับ VB/Win 4.0 ตามลำดับได้ทันที เพราะว่าคอนโทรลเหล่านี้มีความคอมแพคทิเบิลกัน ยกเว้นเพียงแต่คอนโทรล Rich Text เท่านั้นที่เป็นคอนโทรลระดับ 32 บิต (RICHTX32.OCX) ซึ่งเป็นคอนโทรลที่มีความสามารถด้านการแสดงผล เช่น สามารถแสดงผลข้อความได้ด้วยฟอนต์หลาย ๆ รูปแบบและหลาย ๆ ขนาดต่าง ๆ กัน และขนาดของข้อความที่สามารถแสดงผลได้ภายในคอนโทรล เหนือกว่าคอนโทรล Text Box มากทีเดียว และนอกจากนี้คอนโทรล Rich Text ยังสนับสนุนการอ่านและจัดเก็บไฟล์ในรูปแบบ .RTF อีกด้วย ดังตัวอย่างในรูปที่ 7 ผู้เขียนได้ใช้คอนโทรล Rich Text ในการเปิดไฟล์ PASSWORD.RTF โดยการใช้วิธี drag & drop ชื่อไฟล์ดังกล่าวจาก Windows Explorer เป็นต้น

ความสอดคล้องกับระบบ

เนื่องจากวินโดวส์เป็นระบบปฏิบัติการที่สนับสนุนอุปกรณ์และความละเอียดในการแสดงผลของจอภาพมากมาย ดังนั้นแอพพลิเคชันต้องสนับสนุนการแสดงผลอย่างถูกต้องที่ระดับความละเอียดต่าง ๆ ซึ่งผู้อ่านสามารถอ่านระดับความละเอียดหรือการติดตั้งดีไวซ์ต่าง ๆ ของวินโดวส์ได้โดยใช้ฟังก์ชันวินโดว์ API เป็นต้น ซึ่งก็เช่นเดียวกับ VB/Win 4.0-32 บิต ที่มาพร้อมกับคอนโทรล System Info (มีเฉพาะ VB/Win 4.0 ที่มากับ CD-ROM เท่านั้น) ซึ่งเป็นคอนโทรลที่ช่วยในการตรวจจับความเปลี่ยนแปลงต่าง ๆ ของระบบ ซึ่งอาจจะเกิดขึ้นจากการเปลี่ยนแปลงของผู้ใช้ในขณะที่โปรแกรมกำลังทำงานก็ได้ ดังนั้นคอนโทรลดังกล่าวจึงสามารถช่วยให้โปรแกรมเมอร์สามารถสร้างแอพพลิเคชันที่สามารถตอบสนองต่อการเปลี่ยนแปลงของระบบได้เป็นอย่างดี และนอกจากนี้การออกแบบส่วนที่ใช้อินเทอร์เฟสกับผู้ใช้ก็เป็นจุดหนึ่งที่ไมโครซอฟท์ได้ให้ความสำคัญเป็นอย่างมาก สำหรับแอพพลิเคชันที่ต้องการใช้โลโกของวินโดวส์ 95 ซึ่งได้ถูกกำหนดเป็นข้อแนะนำดังต่อไปนี้

- ในกรณีที่มีการใช้ไดอะล็อกบ็อกซ์แสดงผลอย่างต่อเนื่องมากกว่า 1 (nest display) ก็ให้หันมาใช้ Tab Dialog แทน โดยการใช้คอนโทรล Tab ที่มากับ VB/Win 4.0-32 บิต ในการสร้างไดอะล็อกบ็อกซ์ แบบหลาย ๆ หน้า และแต่ละหน้าก็จะมีชื่อบอกในส่วนป้ายชื่อของ Tab



รูปที่ 7 แสดงตัวอย่างของการใช้คอนโทรล Rich Text แสดงผลไฟล์ PASSWORD.RTF

รูปที่ 8 แสดงตัวอย่างของการใช้ปุ่มขวามือของเม้าส์

- ใช้คอนโทรลที่แสดงผลแบบ 3 มิติ และกำหนดให้ฟอร์มที่ใช้มีการแสดงผลแบบ 3 มิติด้วย (โดยการกำหนดให้คุณสมบัติ Appearance มีค่าเท่ากับ True) ซึ่งจะมีผลต่อการดึงดูดความสนใจและยังมีผลต่อความเป็นมิตรต่อการใช้งานของผู้ใช้อีกด้วย

- กำหนดให้ปุ่มขวามือของเม้าส์มีผลต่อการแสดงผลเมนูหรือรายการคำสั่งที่มีการใช้งานบ่อย โดยต้องสามารถใช้ได้ทุกแห่งภายในฟอร์ม ดังในรูปที่ 8 แสดงให้เห็นตัวอย่างการใช้ปุ่มขวามือของเม้าส์แสดงเมนูเมื่อผู้ใช้คลิ๊กภายในฟอร์ม

การใช้การลงทะเบียนของระบบ

วินโดวส์ 95 มีการลงทะเบียนที่แตกต่างจากวินโดวส์ 3.1x โดยที่วินโดวส์ 3.1x จะมีการใช้ไฟล์ .INI ในการกำหนดคุณสมบัติต่าง ๆ ที่แอพพลิเคชันจำเป็นต้องใช้ หรือต้องการใช้ในการกำหนดความสัมพันธ์กับวินโดวส์ แต่สำหรับวินโดวส์ 95 นั้นไฟล์เหล่านี้ เช่น SYSTEM.INI หรือ WIN.INI เป็นต้น จะถูกแทนที่ด้วยไฟล์การลงทะเบียนของระบบ (system registry) ซึ่งจะเป็นไฟล์ไบนารี่ที่มีการ

จัดการต่างจากไฟล์ .INI โดยตรง แต่เนื่องจาก VB/Win 4.0-32 บิต ก็มีกลุ่มของคำสั่งที่สนับสนุนการจัดการไฟล์ลงทะเบียนโดยตรงอย่างจำกัด ดังนั้นผู้อ่านต้องใช้ฟังก์ชันวินโดว์ API ระดับ 32 บิต สำหรับจัดการงานด้านนี้ (ฟังก์ชันจะถูกจัดเก็บไว้ในไฟล์ SHELL32.DLL และสำหรับวินโดวส์ 3.1x จะถูกจัดเก็บไว้ในไฟล์ SHELL.DLL) โค้ดต่อไปนี้เป็นการประกาศฟังก์ชันเหล่านี้

```
Declare Function RegOpenKey Lib "Shell32" (ByVal HKeyIn As Long, ByVal _
   LPCSTR As String, HKeyOut As Long) As Long
Declare Function RegCloseKey Lib "Shell32" (ByVal HKeyIn As Long) As Long
Declare Function RegQueryValue Lib "Shell32" (ByVal HKeyIn As Long, ByVal _
   SubKey As String, ByVal KeyValue As String, KeyValueLen As Long) As Long
Declare Function RegSetValue Lib "Shell32" (ByVal HKeyIn As Long, ByVal _
   SubKey As String, ByVal lType As Long, ByVal strNewValue As String, _
   ByVal lIngnored As Long) As Long
Declare Sub RegDeleteKey Lib "Shell32" (ByVal HKeyIn As Long, ByVal _
   SubKeyName As String)
```

สำหรับกลุ่มคำสั่งสำหรับทำการอ่านหรือจัดเก็บข้อมูลของแอพพลิเคชันลงในไฟล์ลงทะเบียนระบบ (สำหรับวินโดวส์ 3.1x ฟังก์ชันเหล่านี้จะจัดการกับไฟล์ .INI) นั้น VB/Win 4.0-32 บิต ก็ได้เตรียมไว้ให้ทั้งหมด 4 คำสั่งด้วยกันดังนี้

คำสั่ง	รายละเอียด
GetSetting	อ่านข้อมูลของแอพพลิเคชัน
SaveSetting	จัดเก็บข้อมูลของแอพพลิเคชัน
GetAllSettings	อ่านข้อมูลของแอพพลิเคชัน (ทั้งหมดในส่วนของแอพพลิเคชันที่กำหนด)
DeleteSetting	ลบทิ้งรายการข้อมูลที่กำหนดของแอพพลิเคชัน

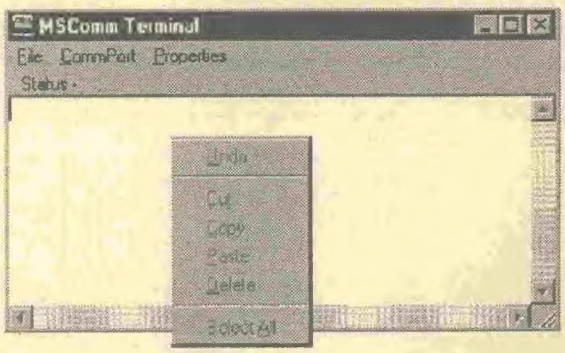
หมายเหตุ ผู้เขียนแนะนำให้ผู้อ่านใช้คำสั่งทั้ง 4 ข้างต้นในการอ่านหรือเขียนข้อมูลไฟล์ลงทะเบียนระบบ ทั้งสำหรับวินโดวส์ 16 บิต และ 32 บิต แทนฟังก์ชันวินโดว์ API ดังเช่น GetPri-vateProfileString หรือฟังก์ชันวินโดว์ API WritePrivateProfileString เป็นต้น ทั้งนี้เนื่องจากฟังก์ชันวินโดว์ API ทั้งสองไม่สนับสนุนสำหรับไฟล์ลงทะเบียนระบบของวินโดวส์ 95

สนับสนุน OLE Automation

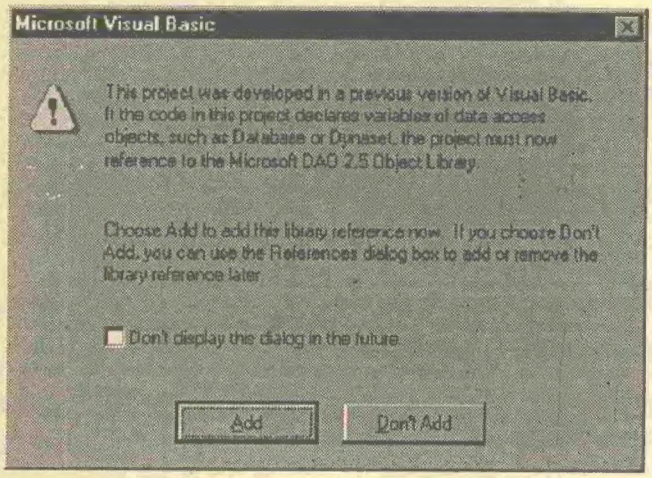
เนื่องจาก OLE Automation เป็นวิธีที่ไมโครซอฟท์ใช้ในการแชร์การควบคุมและออบเจ็กต์ระหว่างแอพพลิเคชัน ดังนั้นเพื่อให้แอพพลิเคชันอื่น ๆ สามารถใช้ความสามารถในส่วนต่าง ๆ ของแอพพลิเคชันที่ผู้อ่านสร้างขึ้นมาได้อย่างเต็มที่ ผู้อ่านก็ควรจะทำการประกาศออบเจ็กต์ (object expose) ที่แอพพลิเคชันสนับสนุน เพื่อให้แอพพลิเคชันอื่น ๆ สามารถสร้างขึ้นใหม่หรือเรียกใช้ออบเจ็กต์ โดยตรงได้อย่างเต็มที่ (การสร้างออบเจ็กต์ สามารถกระทำได้โดยการใช้โมดูล Class และประกาศคลาสนั้น ๆ ด้วยคำสงวน Public) ซึ่งผู้อ่านจะเห็นว่าสำหรับ VB/Win 4.0-32 บิต นี้ไมโครซอฟท์ได้มีการปรับปรุงความสามารถด้าน OLE Automation และ OLE Server ให้ดีขึ้นเป็นอย่างมาก ซึ่งวิธีการสร้างออบเจ็กต์นั้นก็ไม่ยากมากนักและมีลักษณะส่วนใหญ่ใกล้เคียงกับ custom control ของ VB/Win 4.0-32 บิต ซึ่งออบเจ็กต์หนึ่ง ๆ ก็สามารถมีได้ทั้งคุณสมบัติและวิธีสำหรับใช้กำหนดพฤติกรรมหรือวิธีการปฏิบัติการของออบเจ็กต์นั้น ๆ และนอกจากนี้ VB/Win 4.0-32 บิต ยังมีไดอะล็อกบ็อกซ์ Object Browser ที่ช่วยให้การค้นหาและคัดลอกทั้งคุณสมบัติ และวิธีของออบเจ็กต์ ได้อย่างรวดเร็วอีกด้วย ดังตัวอย่างในรูปที่ 9 เป็นไดอะล็อกบ็อกซ์ Object Browser ของแอพพลิเคชันที่ชื่อ Gravity ที่ประกอบด้วยคลาส (ออบเจ็กต์) Ball โดยที่คลาสนี้จะมี 2 คุณสมบัติ และอีก 1 วิธี ส่วน modGravityMain นั้นเป็นเพียงโมดูลมาตรฐาน (.BAS) เท่านั้น (ผู้อ่านสามารถเลือกที่จะดูคุณสมบัติหรือวิธีของโปรเจ็กต์หรือไลบรารีอื่น ๆ ก็ได้ โดยการเลือกชื่อจากช่อง Libraries/Projects:)

สำหรับแอพพลิเคชันที่มีการใช้ความสามารถด้านการจัดการฐานข้อมูลของ Data Access Objects (DAO) เมื่อผู้อ่านโหลดไฟล์โปรเจ็กต์จากเวอร์ชันเก่าของ VB/Win ก็จะปรากฏไดอะล็อกบ็อกซ์ดังในรูปที่ 10 เพื่อให้ผู้อ่านเลือกที่จะเพิ่ม DAO ให้กับแอพพลิเคชัน ซึ่งผมก็ขอให้ผู้อ่านเลือกที่จะเพิ่ม DAO โดยที่ VB/Win 4.0-16 บิต จะใช้ DAO 2.5 ส่วน VB/Win 4.0-32 บิต จะใช้ DAO 3.0 ส่วนสาเหตุที่เราไม่พบออบเจ็กต์ DAO ในไดอะล็อกบ็อกซ์ Custom Controls ก็เพราะว่าไมโครซอฟท์ได้แยก DAO ออกจากออบเจ็กต์อื่น ๆ ที่สามารถเข้าถึงได้โดยใช้ OLE Automation

ถึงจุดนี้ ผู้เขียนก็หวังว่าผู้อ่านคงได้รับความรู้และข้อเสนอแนะสำหรับการสร้างแอพพลิเคชันสำหรับวินโดวส์ 95 มากพอสมควร ซึ่ง



รูปที่ 9 แสดงไดอะล็อกบ็อกซ์ Object Brower ของออบเจ็กต์ Gravity



รูปที่ 10 แสดงไดอะล็อกบ็อกซ์สำหรับการเพิ่ม DAO ให้กับแอพพลิเคชัน

จะช่วยให้ขั้นตอนในการออกแบบหรือวางแผนเกี่ยวกับรูปร่างหน้าตาหรือคุณสมบัติที่แอพพลิเคชันสนับสนุนสั้นลง และได้ผลลัพธ์ที่ถูกต้องมากที่สุด แต่อย่างไรก็ตาม แอพพลิเคชันที่ได้จะสามารถทำงานได้อย่างถูกต้องหรือมีประสิทธิภาพเพียงใด ทั้งนี้ส่วนหนึ่งก็ขึ้นกับประสบการณ์และความสามารถในการเขียนโปรแกรมหรือการประยุกต์อัลกอริทึมที่เหมาะสมมาใช้กับแอพพลิเคชัน ซึ่งสิ่งเหล่านี้ผู้เขียนคงไม่สามารถอธิบายได้หมดโดยอาศัยหนังสือหรือวิธีใด ๆ ก็ตาม เพราะโดยความเป็นจริงมันเป็นสิ่งที่เป็นพรสวรรค์ส่วนหนึ่งและความขยันและโอกาสอีกส่วนหนึ่ง แต่ผู้อ่านสามารถฝึกสิ่งเหล่านี้ขึ้นมาได้เช่นกันด้วยการเขียนแอพพลิเคชันบ่อย ๆ และผู้เขียนก็ขออวยพรให้ผู้อ่านทุกท่านประสพความสำเร็จกับการเขียนโปรแกรมด้วย VB/Win 4.0 ครับ ก่อนจบบทความก็ขอแจ้งให้ผู้อ่านที่ต้องการติดต่อกับผู้เขียนก็ขอให้เขียนจดหมายมาที่หมายเลขตู้ป.ณ.ใหม่ดังนี้ครับ ตู้ป.ณ. 95 ปทฝ.ดินแดง กรุงเทพฯ 10407

เทคนิคประจำคอลัมน์

ถาม ประโยค For Each ... In ... Next ของ VB/Win 4.0 นั้น มีวิธีการใช้อย่างไร ?

ตอบ เนื่องจาก VB/Win ที่สนับสนุนการโปรแกรมตามแนวความคิดของ OOP ดังนั้น จึงประกอบด้วยออบเจ็กต์ ต่าง ๆ มากมาย โดยเฉพาะที่มีชื่อเรียกสำหรับ VB/Win ว่า custom control (.OCX, .VBX) นั้นเป็นออบเจ็กต์ที่ช่วยลดภาระการเขียนโปรแกรมได้มากทีเดียว ซึ่งในแต่ละฟอร์มเราสามารถที่จะใส่คอนโทรลเหล่านี้ได้มากมาย ดังนั้น VB/Win 4.0 จึงได้ทำการสร้างคำสั่งที่ช่วยในการค้นหาคอนโทรลในคอลเลคชัน (collection) เป็นการเฉพาะเช่นเดียวกับการวนลูป ดังโค้ดต่อไปนี้เป็นการค้นหาเฉพาะคอนโทรล TextBox และทำการกำหนดให้คุณสมบัติ Text มีค่าเท่ากับVisual Basic 4.0 ดังนี้

```
Public Sub gClearForm(frmName As Form)
   Dim ctrl As Control

   For Each ctrl In frmName.Controls
     If TypeOf ctrl Is TextBox Then
       ctrl.Text = "Visual Basic 4.0"
     End If
 Next ctrl
End Sub
```

ซึ่งผู้อ่านจะเห็นว่าโค้ดที่ได้จะมีความกระชับและง่ายต่อการทำความเข้าใจกว่า การใช้ประโยค For...Next ในการค้นหาคอนโทรล TextBox เพราะประโยค For...Next ไม่สนับสนุนการเข้าถึงแต่ละสมาชิกของ คอลเลคชันโดยตรง